Day 3 :

Inheritance:

Single Inheritance:

#include <iostream>

using namespace std;

class Account

{

public:

float salary = 60000;

};

class Programmer: public Account

{

public:

float bonus = 5000;

};

int main(void)

{

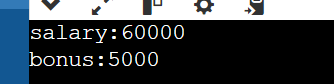
Programmer p1;

cout<<"salary:"<<p1.salary<<endl;

cout<<"bonus:"<<p1.bonus<<endl;

return 0;

}



Multilevel Inheritance:

#include <iostream>

using namespace std;

class Animal

{

public:

void eat()

{

cout<< "eating..."<<endl;

}

};

class Dog: public Animal

{

public:

void bark()

{

cout<<"barking..."<<endl;

}

};

class BabyDog: public Dog

{

public:

void weep()

{

cout<<"weeping..."<<endl;

}

};

int main(void)

{

BabyDog d1;

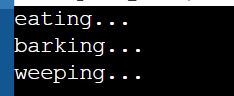
d1.eat();

d1.bark();

d1.weep();

return 0;

}



Multiple Inheritance

#include<iostream>

using namespace std;

class A

{

protected:

int a;

public:

void get\_a(int n)

{

a = n;}

};

class B

{

protected:

int b;

public:

void get\_b(int n)

{

b = n;

}

};

class C : public A,public B

{

public:

void display()

{

cout<< "The value of a is:"<< a << std::endl;

cout<< "The value of b is:"<< b << std::endl;

cout<< "addition of a and b is:"<< a+b << std::endl;

}

};

int main()

{

C c;

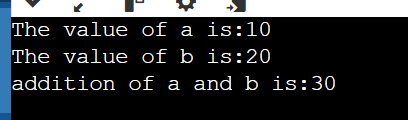
c.get\_a(10);

c.get\_b(20);

c.display();

return 0;

}



Hybrid Inheritance:

#include<iostream>

using namespace std;

class A

{

protected:

int a;

public:

void get\_a()

{

cout<<"enter the value of a : "<<endl;

cin>>a;

}

};

class B : public A

{

protected:

int b;

public:

void get\_b()

{

cout<<"enter the value of b : "<<endl;

cin>>b;}

};

class C

{

protected:

int c;

public:

void get\_c()

{

cout<< " enter the value of c is:" <<endl;

cin>>c;

}

};

class D : public B , public C

{

protected:

int d;

public:

void mul()

{

get\_a();

get\_b();

get\_c();

cout<<"multiplication of a,b,c is:"<<a\*b\*c<<endl;

}

};

int main()

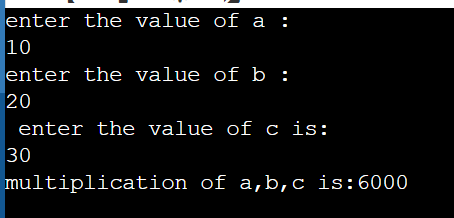
{

D d;

d.mul();

return 0;

}



Program:

#include <iostream>

using namespace std;

class A

{

public:

void display()

{

std::cout<<"class A"<<std::endl;

}

};

class B

{

public:

void display()

{

std::cout<<"class B"<<std::endl;

}

};

class C : public A,public B

{

public:

void view()

{

A::display();

B::display();

}

};

int main()

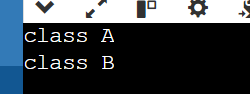
{

C c;

c.view();

return 0;

}



Function overloading:

#include <iostream>

using namespace std;

class cal

{

public:

static int add(int a,int b)

{

return a + b;

}

static int add(int a ,int b, int c)

{

return a + b + c;

}

static int sub(int a,int b)

{

return a - b;

}

static int sub(int a,int b,int c)

{

return a - b - c;

}

static int divide(int a,int b)

{

return a/b;

}

static int divide(int a, int b, int c)

{

return a/b/c;

}

static int multiply(int a,int b)

{

return a \* b;

}

};

int main(void)

{

cal C;

cout<<"the addition of two numbers is: "<<C.add(10,20)<<endl;

cout<<"the addition of three numbers is: "<<C.add(12,20,23)<<endl;

cout<<"the subtraction of two numbers is: "<<C.sub(30,20)<<endl;

cout<<"the subtraction of three numbers is: "<<C.sub(60,20)<<endl;

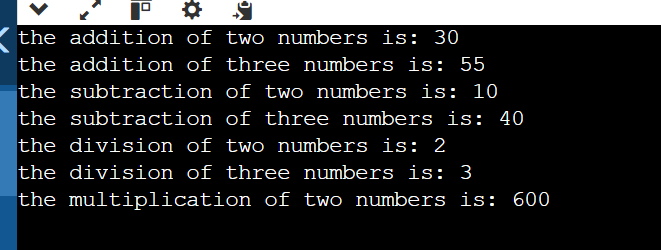
cout<<"the division of two numbers is: "<<C.divide(60,30)<<endl;

cout<<"the division of three numbers is: "<<C.divide(150,50)<<endl;

cout<<"the multiplication of two numbers is: "<<C.multiply(30,20)<<endl;

return 0;

}



Program:

Design a class hierarchy to represent various entities in a university system.

Base Class (Person):

Data members: name (string), age (int)

Member functions: getDetails(), a virtual function to print basic person details

Derived Class (Student): (Single Inheritance)

#include <iostream>

#include <string>

#include <vector>

class Person

{

protected:

std::string name;

int age;

public:

Person(const std::string& name, int age) : name(name), age(age) {}

virtual std::string getDetails() const

{

return "Name: " + name + ", Age: " + std::to\_string(age);

}

virtual void printDetails() const

{

std::cout << getDetails() << std::endl; }

virtual ~Person() = default;

};

class Student : public Person

{

protected:

int studentId;

std::string department;

public:

Student(const std::string& name, int age, int studentId, const std::string& department)

: Person(name, age), studentId(studentId), department(department) {}

void setMajor(const std::string& department)

{

this->department = department;

}

std::string getDepartment() const

{

return department;

}

std::string getDetails() const override

{

return Person::getDetails() + ", Student ID: " + std::to\_string(studentId) + ", Department: " + department;

}

void printDetails() const override

{

std::cout << getDetails() << std::endl; }

};

int main()

{

Person person("Vidya", 23);

person.printDetails();

Student student("Divya", 24, 12345, "Computer Science");

student.printDetails();

return 0;

}

